Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

Please amend Claims 1, 4, 6, 7, 9-11, 13, 19-21, and 27, cancel Claims 15-18 and 23-26, and add new Claim 28, as follows.

1. (Currently Amended) A method for authenticating an external module comprising the steps of:

~~creating~~ providing data K that can be created by using two different schemes, at least one scheme of the two different schemes being based on the integrity of ~~the~~ a module to be verified;

~~creating~~ providing an authentication token for said module which produces data K in both schemes, an executable external module being representative of said module;

at a computer system, using data K as created by one scheme of the two different schemes to disrupt said executable external module; and

at the computer system, using data K as created by the other scheme of the two different schemes to restore the executable external module from the disrupted executable external module thereby authenticating said executable external module.

2. (Original) The method as defined in claim 1, wherein one or more of the schemes is based on RSA encryption.

3. (Original) The method as defined in claim 1, wherein one or more of the schemes is based on digital signets.

4. (Currently Amended) The method as defined in claim 1, further comprising the step of:

embedding a public component of one or more of the schemes in a verification code of the executable external module. ~~application~~.

5. (Original) The method as defined in claim 1, further comprising the step of:

conveying private components of all schemes to a module authentication authority.

6. (Currently Amended) The method as defined in claim 5, wherein the ~~creating~~ providing an authentication token step is automated by providing developers of the ~~external~~ module access to a Web site which allows them to submit the hash of their module and to retrieve the authentication token.

7. (Currently Amended)The method as defined in claim 1, wherein the authentication token is embedded

in the executable external module as long as the executable external module itself remains functional.

8. (Original) The method as defined in claim 7, where the embedding is realized by adding an

additional data section to a DLL in Portable Executable (PE) format.

9. (Currently Amended)The method as defined in claim 1, wherein the authentication token is external to the executable external module.

10. (Currently Amended) The method as defined in claim 1, wherein said at least one scheme depending on code integrity is independent of a location of the executable external module in memory.

11. (Currently Amended) The method as defined in claim 10, wherein location independence is achieved by locating and reading the executable external module's image on a disk.

12. (Original) The method as defined in claim 10, wherein location independence is achieved by using a canonical hash.

13. (Currently Amended) A method for secure authentication of external modules on an entity comprising the steps of:

~~loading~~ storing an executable external module in ~~into~~ memory at a computer system; ~~and~~

~~beginning a~~ STOMPing, at the computer system, the executable external module ~~process~~ by

decrypting a number of pseudo-random bytes that are part of an authentication token using a public security code of a public and private component pair security code; and

XORing the decrypted pseudo-random bytes with the executable external module thereby disrupting the executable external module into an unusable state; and

UNSTOMPing, at the computer system, the executable external module in the unusable state by

performing a signet extrication process to generate extrication data by using the hash of the executable external module in the unusable state;

using the extrication data to generate another stream of pseudo-random bytes; and

XORing the another stream of pseudo-random bytes with the executable external module in the unusable state thereby

restoring, from the executable external module in the unusable state, the executable external module back to a usable state in the event there has been no illicit patching of the executable external module, and

maintaining the executable external module in an unusable state in the event that the executable external module has been illicitly patched such that an application or program that is accessing the executable external module in the unusable state fails to operate.

14. (Original)The method as defined in claim 13, wherein the public and private components of

the security code comprise security codes selected from a group of security codes of: a signet pair and an RSA pair.

Claims 15-18. (Canceled)

19. (Currently Amended) The method as defined in claim 13 ~~18~~, further comprising the step of:

re-authenticating, at the computer system, the executable external module by periodically performing the STOMP and UNSTOMPing process multiple times while interacting with the executable external module at the computer system.

20. (Currently Amended) The method as defined in claim 13 ~~18~~, further comprising the step of:

performing run time checks of the executable external module to make sure that function calls to the executable external module are not intercepted by an attacker.

21. (Currently Amended) A computer readable medium comprising instructions for secure authentication of

executable external modules at a computer system ~~on an entity~~ comprising the instructions of:

~~loading~~ storing an executable external module in ~~into~~ memory at a computer system; ~~and~~

~~beginning a~~ STOMPing, at the computer system, the executable external module ~~process~~ by

decrypting a number of pseudo-random bytes that are part of an authentication token using a public security code of a public and private component pair security code; and

XORing the decrypted pseudo-random bytes with the external module thereby disrupting the executable external module into an unusable state; and

UNSTOMPing, at the computer system, the executable external module in the unusable state by

performing a signet extrication process to generate extrication data by using the hash of the executable external module in the unusable state;

using the extrication data to generate another stream of pseudo-random bytes; and

XORing the another stream of pseudo-random bytes with the executable external module in the unusable state thereby

restoring, from the executable external module in the unusable state, the executable external module back to a usable state in the event there has been no illicit patching of the executable external module, and

maintaining the executable external module in an unusable state in the event that the executable external module has been illicitly patched such that an application or program that is accessing the executable external module in the unusable state fails to operate.

22. (Original) The computer readable medium as defined in claim 21 wherein the public and

private components of the security code comprise security codes selected from a group of security codes of: a signet pair and an RSA pair.

Claims 23-26. (Canceled)

27. (Currently Amended) A computer system for secure authentication of executable external modules, ~~on an entity~~ comprising ~~the units of~~:

a loader unit for loading an executable external module into memory; ~~and beginning a STOMPing process by~~

a decryption unit for decrypting a number of pseudo-random bytes that are part of an authentication token using a public security code of a public and private component pair security code; and

a processor, communicatively coupled with the memory and the decryption unit, for

XORing the decrypted pseudo-random bytes with the executable external module thereby disrupting the executable external module into an unusable state;

performing a signet extrication process to generate extrication data by using the hash of the executable external module in the unusable state;

using the extrication data to generate another stream of pseudo-random bytes; and

XORing the another stream of pseudo-random bytes with the executable external module in the unusable state thereby

restoring, from the executable external module in the unusable state, the executable external module back to a usable state in the event there has been no illicit patching of the executable external module, and

maintaining the executable external module in an unusable state in the event that the executable external module has been illicitly patched such that an application or program that is accessing the executable external module in the unusable state fails to operate.

28. (New) The method as defined in claim 1, wherein

the using data K as created by one scheme to disrupt said executable external module, and

the using data K to restore the executable external module from the disrupted executable external module,

are performed at run time of the executable external module at the computer system.